

# An objective comparison test of workload management systems

Igor Sfiligoi<sup>1</sup> and Burt Holzman<sup>1</sup>

<sup>1</sup>Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

E-mail: sfiligoi@fnal.gov

**Abstract.** The Grid resources are distributed among hundreds of independent Grid sites, requiring a higher level Workload Management System (WMS) to be used efficiently. There are several ways to design and implement a WMS, and indeed in recent years several WMSes have been developed. The purpose of this paper is to show how some of these different WMSes behave under realistic load conditions. We present benchmark test results for three general-purpose WMSes, namely ReSS, gLite WMS and glideinWMS. The results presented have been measured using the same tools for all the tested WMSes, comparing those results against a baseline obtained by using plain Condor-G submissions.

## 1. Introduction

With millions of jobs coming from thousands of users, and tens of thousands resources deployed to run them, a higher level Workload Management System (WMS) is essential for the Grid[1] be used efficiently. However, there are several ways to design and implement a WMS, and indeed in recent years several WMSes have been developed. Most of them have been developed by groups for their own specific needs, but there are a few that claim to be general enough to serve a wide variety of users.

Having multiple products to choose from is obviously a major advantage for the users. But objective data is needed to help users choose the right one. In this paper we present benchmark test results for four of them, namely Condor-G, ReSS, gLite WMS and glideinWMS. The latest version available at the end of August 2007 was used for all the WMSes.

The test suite used was developed locally specifically for these tests. It uses the same benchmarking logic for all of the tested WMSes, with WMS-specific plugins for job submission and job monitoring. Only pre-WS GRAM[1] grid submissions have been tested, as we have the most experience with it.

A summary functionality overview of the WMSes is also given, but it is by no means complete. WMS-specific documentation should be consulted for more detailed information about the product functionality

## 2. Description of Workload Management Systems

### 2.1. Condor-G

Condor[2] is a multi-featured batch system provided by the Condor team at the University of Wisconsin-Madison. It contains a submission mode called “Condor-G”, which provides a translation layer that presents grid resources in a manner similar to that of a local batch system. It translates Condor job description files into grid specific commands, handling job submission, staging of input and output, and monitoring. It submits multiple grid submission protocols, including among others the pre-WS GRAM, WS-GRAM and Nordugrid.

Condor-G is used both for direct submission from final users, and as the underlying submission mechanism by many WMSes. In this work, we used Condor-G to determine a raw baseline.

### 2.2. ReSS

The Resource Selection Service (ReSS)[3] is an OSG endorsed matchmaking system for jobs submitted via Condor-G. It allows users to specify the target grid compute elements (CEs) in terms of logical attributes, like the operating system, and ReSS will make the actual decision. Once a decision is taken, the job is submitted by the local Condor-G itself.

ReSS collects information about CEs by subscribing to the CEMon service, deployed on most OSG CEs. OSG also provides a centrally-managed ReSS information system, that can be used to populate a local one. Support for other grids is currently not envisioned.

### 2.3. gLite WMS

The gLite Workload Management System (gLite WMS)[4] is a EGEE endorsed product. It is a portal solution -- users submit their jobs to a WMS server using dedicated gLite-specific client tools, and the WMS handles submission, resource brokering, input/output staging, and job monitoring internally. Users specify the target grid compute elements (CEs) in terms of logical attributes, like in the ReSS case.

gLite WMS can collect information about CEs in a variety of ways. The most commonly used method is based on the BDII service, deployed both on EGEE and OSG CEs.

gLite WMS can internally use various job submission mechanisms, but Condor-G is used for pre-WS GRAM based CEs.

### 2.4. glideinWMS

The glideinWMS[5] was developed by the CMS collaboration based on the previous work done by the CDF collaboration, but it is meant to be general enough to be used outside the high energy physics domain. It is based on the pilot philosophy and uses Condor as the base batch system.

In glideinWMS, users submit to a local Condor scheduler, and are never exposed to the grid. Instead, pilot jobs containing Condor daemons are submitted to various grid CEs. Once a pilot job stats, the local environment is discovered, and that information is used to select the appropriate user job. From the user point of view, it is like running in a virtual private local pool.

The information about the grid CEs is statically configured by the glideinWMS administrator; the administrator itself can use any other information system, like the above mentioned CEMon and BDII services.

The pilot jobs are submitted using Condor-G.

### 3. Test results

CMS has set up shadow pools at a few of its Open Science Grid (OSG) production grid sites. A shadow pool is a batch system pool overlaid over an existing production pool. While the worker nodes are the same for both pools, the submission point and the control services are separate.

In order not to interfere excessively with the production pool, only sleep jobs with a small sandbox were being run on those shadow pools for the purpose of these tests.

#### 3.1. Condor-G

Condor-G showed significant slowdown when more than approximately 7000 jobs are in the queue. So all results in table 1 were obtained by keeping below that limit. In this setup, we were easily able to saturate our test grid CE with its 5000 batch slots.

While we had no problem submitting 20000 jobs, when the threshold of 7000 was exceeded, Condor-G started to use 100% of the available CPU and the submission rate to the test CE was reduced to almost zero. Indeed, letting it run for a long time, the system stabilized with just 200 running jobs on the grid side.

Condor-G has a lot of configuration parameters, so we decided to try them out. The one that really makes a difference is `GRIDMANAGER_MAX_JOBMANAGERS_PER_RESOURCE`. The default is only 10, so we repeated our tests by setting it to 100; the results are shown in table 2. As can be seen, the remote job rates doubled. Unfortunately, the load on the test grid CE increased by a factor 10! And while trying to remove 5000 running jobs at once, we effectively killed the CE; we had to manually wipe out the jobs, after trying to stabilize it for a full day, without success.

Querying the queue about job status was easy and fast. With 20000 jobs in the queue, the query request returned within a second.

**Table 1.** Condor-G with default settings

Test	Rate (per minute)
Job submission rate	250
Job startup rate	30
Local removal rate	O(10000)
Remote removal rate	30
Query rate	O(10000)

**Table 2.** Condor-G with

`GRIDMANAGER_MAX_JOBMANAGERS_PER_RESOURCE = 100`

Test	Rate (per minute)
Job submission rate	250
Job startup rate	60
Local removal rate	O(10000)
Remote removal rate	60
Query rate	O(10000)

Job success rate varied significantly among the sites. On well behaving sites, all the jobs ran fine and returned a significant output; on misconfigured sites, all the jobs may fail, and no output be returned. And then you have the intermediate case, when only a fraction of the jobs fail.

The major problem is identifying bad sites; once we tested the same site twice in two days, and one day it was working perfectly fine, the next it was completely broken. It turned out that the grid

administrator has changed a configuration parameter in a wrong way, and that broke the grid CE. Unfortunately, there was no way a user to know this, without actually trying to run some jobs.

### 3.2. ReSS

The tests we performed with ReSS showed exactly the same behavior as the Condor-G tests shown in table 1. Query times and job success rates were also essentially the same.

No ReSS specific limitations were found. However to be correct, by using a single type of jobs, ReSS resource matching capabilities were not really stressed.

### 3.3. gLite WMS

The initial gLite WMS submission tests proved that it cannot be used in single submission mode. We were able to submit only 100 jobs in 20 minutes. We did not perform any other tests in this mode, as we deemed this mode unusable for any practical purpose.

gLite WMS supports a so called “collection mode” submission. In this mode several jobs are submitted in a single transaction. This indeed speeds up the submission rate quite a bit, as we were able to submit a collection of 5000 jobs in 5 minutes, while a 20000 job collection took 23. However, trying to submit several collections in a short amount of time proved to be tricky; we were never able to submit 8 consecutive collections of 5000 jobs each in a row, and 8 parallel submissions of same sized collections resulted in only 1 of them being submitted, and all the others failing. By putting a few minute delay between submissions, 4 collections of 20000 jobs each did succeed.

Querying the gLite WMS proved to be a frustrating experience, too. We found no easy way to obtain the list of all the jobs that we owned on the portal. Moreover, the queries often failed, claiming the portal was overloaded at that particular moment. When the queries did succeed, they returned the status of a 5000 job collection in 40 seconds and the status of 20000 job collection in 3 minutes.

Removal speeds depended on portal load. The fastest we observed was a 20000 job collection being removed in 30 seconds, the worse was a 5000 job collection being removed in 1 minute. Both of those numbers are for local removal only; we have no numbers for how long would it take to remove the jobs from the grid CE queues.

Similarly, we have no numbers regarding the startup times on the grid CE side; we were unable to interface our test pool with the gLite WMS instance we were using, in the time allotted for the tests.

The above results are summarized in table 3.

**Table 3.** gLite WMS results

Test	Rate (jobs per minute)
Job submission rate (single mode submission)	5
Job submission rate (bulk mode submission)	O(1000)
Local removal rate	100-400
Query rate	100

Job success rates were essentially the same as with Condor-G and ReSS.

### 3.4. glideinWMS

We set up glideinWMS with each daemon running on a separate machine, and all the measurements were taken on the submit machine.

The results are presented in table 4. The only interesting point regards the startup rates; job startup in a freshly installed system is significantly slower than job startup in a running system. The reason for this disparity lies in the underlying pilot submissions; since Condor-G is used for pilot submission, they cannot start faster than what Condor-G provides. But when the pilots are running, they can start several jobs much faster.

We were able to queue 80000 jobs in the queue with no noticeable degradation of performance, and up to 5000 jobs were seen running. Two limits have been observed:

- Each running job uses 1.3MB of memory on the submit machine. Our 8GB machine could not have handled more than 5000 running jobs.
- When GCB was needed, due to firewalls or NATs, the system was stable only to about 600 pilots per GCB.

Both of the above problems could be addressed by installing more machines, but are real limits if more hardware is not an option.

Job queries were easy and fast as with Condor-G, but much more information was provided about the jobs' status. The queries provided the exact node name where the job was running, as well as a wealth of information about the node itself, like available memory and disk space, updated every few minutes with real time values.

**Table 4.** glideinWMS results

Test	Rate (jobs per minute)
Job submission rate	450
Job startup rate (cold start)	30
Job startup rate (steady state)	200
Local removal rate	O(10000)
Remote removal rate	120
Query rate	O(10000)

Job success rate was above 99.99% rate, even in presence of misbehaving grid resources. In case of a problem with a grid resources, either the pilot would fail before starting a job, or the job would be restarted on a different node, if the job was killed after starting.

## 4. Summary

The four WMSes we tested take each quite a different approach to solving the job management problem, and each has its strong points and its disadvantages. See table 5 and table 6 for an overview.

**Table 5.** Tested WMS installation comparison

WMS	Site selection	Client	Server
Condor-G	User	Light daemon	None
ReSS	CEMon	Light daemon	One light daemon node
gLite WMS	BDII, CEMon	None	Two heavy daemon nodes
glideinWMS	Admin	Heavy daemon	Several heavy daemon nodes

**Table 6.** Tested WMS scalability limits and functionality

WMS	Single client scalability	Server scalability	Functionality
Condor-G	Up to 7000 queued jobs	N/A	Just submission
ReSS	Up to 7000 queued jobs	None observed	Submission and site selection
gLite WMS	Requires grouping of jobs in big collections	Suffers with 20000 queued jobs, but no limits found when using retries	Portal solution, site selection
glideinWMS	Memory limited, 1.3MB per running job	None found, but needs one GCB server for every 600 running jobs	Pilot solution: <ul style="list-style-type: none"> <li>• just-in-time scheduling</li> <li>• node validation</li> <li>• active job management</li> </ul>

We thus make no recommendations.

### Acknowledgements

The authors gratefully acknowledge the support of the CMS experiment and the Department of Energy. We also would like to thank Terrence Martin (UCSD), Michael Thomas (Caltech), and Daniele Cesini (CNAF) and their respective institutions for allowing us the use of their resources.

### References

- [1] Foster I and Kesselman C 1998 *The Grid: Blueprint for a New Computing Infrastructure*. (San Francisco, CA: Morgan Kaufmann Publishers)
- [2] Thain D, Tannenbaum T and Livny M 2005 Distributed Computing in Practice: The Condor Experience *Concurrency - Practice and Experience* **17** 2-4 323-56
- [3] <https://twiki.grid.iu.edu/twiki/bin/view/ResourceSelection/WebHome>
- [4] <http://glite.web.cern.ch/glite/packages/R3.0/deployment/glite-WMS/glite-WMS.asp>
- [5] <http://home.fnal.gov/~sfiligoi/glideinWMS/>